

**操作系统原理课程设计报告**

|  |  |
| --- | --- |
| 姓 名： |  |
| 学 院： |  |
| 专 业： |  |
| 班 级： |  |
| 学 号： |  |
| 指导教师： |  |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

年 月 日

目录

[1 实验一 文件拷贝和并发程序 1](#_Toc98859941)

[1.1 实验目的 1](#_Toc98859942)

[1.2 实验内容 1](#_Toc98859943)

[1.3 实验设计 1](#_Toc98859944)

[1.3.1 开发环境 1](#_Toc98859945)

[1.3.2 实验设计 1](#_Toc98859946)

[1.4 实验调试 2](#_Toc98859947)

[1.4.1 实验步骤 2](#_Toc98859948)

[1.4.2 实验调试和心得 2](#_Toc98859949)

# 实验一 文件拷贝和并发程序

## 实验目的

掌握Linux操作系统的使用方法，包括键盘命令、系统调用；掌握在Linux下的编程环境。

## 实验内容

1. 编一个C程序，其内容为实现cp/文件拷贝的功能(使用系统调用open/read/write，实现cp命令)；
2. 编一个C程序，其内容为分窗口同时显示三个并发进程的运行结果。要求用到Linux下的图形库(gtk/Qt)，如三个进程实现誊抄的演示。

## 实验设计

### 开发环境

**物理机环境：**

处理器：Intel® Core™ i7-10870H CPU @ 2.20GHz

操作系统：Windows10专业版 21H2 (内部版本19044.1586)

**虚拟机环境：**

虚拟化软件：VMware® Workstation 16 Pro 16.2.1 build-18811642

虚拟系统镜像：Ubuntu 16.04.7 desktop amd64

### 实验设计

1. 要实现文件复制，只需要循环从源文件中读取一段字符串再将其写入到目标文件中即可。具体实现中，每次读取1024个字符。当某次读取到的字符数少于1024，表明读取到了文件结尾，此时退出循环并将这些字符写入到目标文件中。可利用system函数调用md5sum命令计算源文件和目标文件的md5值来判断复制后的文件是否与源文件相同。
2. 要实现分窗口显示三个并发进程的运行结果，可以利用QT提供的label组件编写三个带窗口的程序，随后在一个主进程中创建子进程并用execv命令调用这三个程序即可。实验中，三个子进程的功能分别为：循环显示数字0~9、显示当前系统时间和循环显示英文字母A~Z。

## 实验调试

### 实验步骤

1. 按照1.3.2（1）中的思路编写程序，使用gcc编译后运行即可。可使用diff命令比较源文件和目标文件是否有差异，或者分别计算两文件的md5值。
2. 编写1.3.2（2）中描述的的三个子程序，其中子进程1，3设置为0.5秒刷新一次，子进程2每1秒刷新一次，在QT中编译出可执行文件。编写主进程，创建三个子进程并在其中使用execv调用生成的三个可执行文件，使用gcc编译并运行即可。

### 实验调试和心得

对于任务1，使用gcc将源代码编译成可执行文件，运行后使用diff命令和计算md5的方法对目标文件进行比较

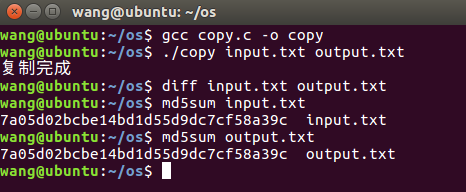


图 1.1 程序的编译运行和结果正确性判断

对于任务2，将主程序和三个子程序放在同一目录下，直接运行主程序，可观察到出现三个窗口，内容分别是三个子程序的输出。

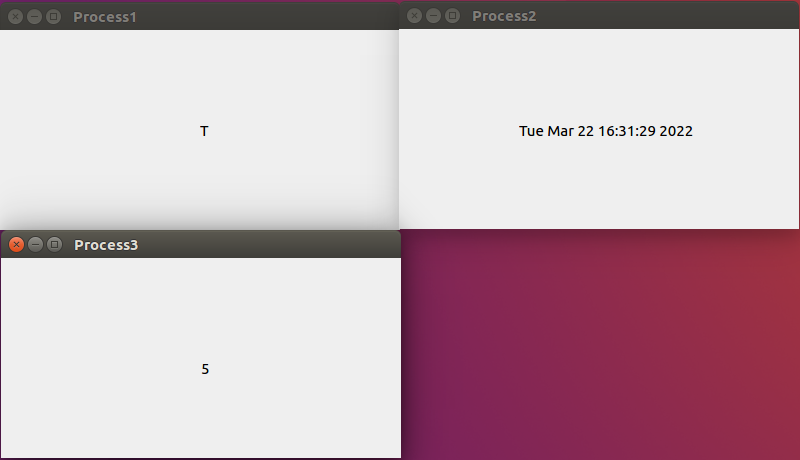


图 1.2 三个并发进程的展示结果

**心得体会：**

任务一内容较为容易，实现文件的复制只需要利用到简单的文件读写操作。需要注意的是读取缓冲区较大时对于结束条件的判断。另外判断输入输出文件是否相同需利用diff或md5计算的方法进行，而不宜直接观察。

任务二的难点在于图形界面的使用。QT提供了多种组件以及完善的组件操作，但是熟悉这些组件和操作需要花费一定的时间。此外，为实现窗口内容的定时更新，我们需要在每次刷新后将程序暂停一段时间，而使用sleep函数会导致报错。因此我们需要通过绑定计时器的方式来自己实现sleep函数的功能。

# 实验二 新增系统调用

## 实验目的

掌握系统调用的实现过程。

## 实验内容

通过编译内核方法，增加一个新的系统调用，实现的功能是文件拷贝。

另编写一个应用程序，调用新增加的系统调用。

## 实验设计

### 开发环境

**物理机环境：**

处理器：Intel® Core™ i7-10870H CPU @ 2.20GHz

操作系统：Windows10专业版 21H2 (内部版本19044.1586)

**虚拟机环境：**

虚拟化软件：VMware® Workstation 16 Pro 16.2.1 build-18811642

虚拟系统镜像：Ubuntu 16.04.7 desktop amd64

原内核版本：4.19.230

替换内核版本：4.14.268

### 实验设计

按照1.3.2（1）的思路，编写实现文件拷贝功能的函数，但是在这里，我们无法使用实验一中使用的库函数，我们需要将这些函数修改为sys\_开头的函数。

要将代码编译进linux内核中，我们需要对linux内核源码进行修改。首先打开linux源码文件夹，打开/kernel/sys.c文件，将编写好的源代码添加到其中。随后，在/arch/X86/include/asm/syscalls.h中，加入编写的函数的声明。最后在/arch/X86/entry/syscalls/syscall\_64.tbl中为新的系统调用分配系统调用号即可。

## 实验调试

### 实验步骤

1. 编写代码和修改源文件

编写实现文件拷贝功能的系统调用函数，将其命名为sys\_mycopy，按照2.3.2所示步骤对内核源文件进行修改。其中，系统调用好333~511均未被使用，可将系统调用号设置为333。

1. 生成配置文件

在linux目录文件夹下，使用make menuconfig命令，生成编译时的.config配置文件。由于我们不需要进行额外的设置，可直接保存并退出。

1. 编译和安装内核

输入命令make -jN即可开始编译，其中N为线程数。由于我为虚拟机分配了8个虚拟核心，此处N值可设为8。编译完成后，输入命令make modules\_install安装模块，最后输入命令make install安装编译好的内核。

1. 重启和选用新内核

重启虚拟机，在出现提示时迅速按下Shift按键进入Grub界面，在Advanced options for Ubuntu下选择编译安装好的内核即可选中并进入系统。

1. 编写和运行测试程序

使用syscall函数和为其分配的系统调用号（333）进行系统调用，观察文件是否复制成功、复制是否正确。

### 实验调试和心得

使用4.19.230版本内核编译完毕后，在安装模块时，出现文件缺失的错误

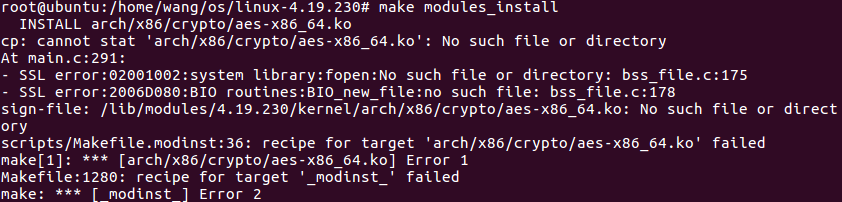


图 2.1 确实模块报错

进行单线程编译，观察到sys\_open，sys\_close，sys\_read，sys\_write四个函数报错，出现了函数未声明的错误，导致模块文件缺失。经网络搜索后未能解决问题，更换内核源码版本为4.14.268后未出现此问题。

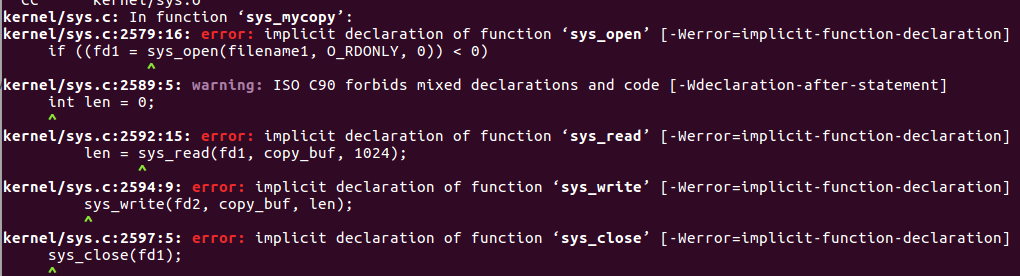


图 2.2 四个函数的未声明错误

编译安装成功后，重启并选择编译的内核，运行测试程序。测试程序中使用system函数调用md5sum命令计算了输入文件和输出文件的md5值，可见，复制成功且结果正确。

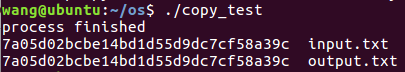


图 2.3 测试程序的运行结果

**心得**：Linux内核的编译是一个耗时很长的工作，因此我们会采用多线程编译的方式来加快这一过程。但是在实际过程中，即使编译过程中一个线程出现问题停止，其他线程仍会继续运行而不是报错，结束时也未输出报错信息，这导致较难发现问题所在。最初出现问题时，我以为是增加系统调用操作不对而进行了多次重新操作，却总是出现相同的问题。最后采纳网上的建议进行单线程编译才发现问题所在。因此对于多线程编译出现的错误，我们可以在单线程下重新运行来定位问题所在。

# 实验三 增加字符设备驱动

## 实验目的

掌握增加设备驱动程序的方法。

## 实验内容

通过模块方法，增加一个新的设备驱动程序，其功能可以简单。

实现字符设备的驱动，演示简单字符键盘缓冲区或一个内核单缓冲区。

## 实验设计

### 开发环境

**物理机环境：**

处理器：Intel® Core™ i7-10870H CPU @ 2.20GHz

操作系统：Windows10专业版 21H2 (内部版本19044.1586)

**虚拟机环境：**

虚拟化软件：VMware® Workstation 16 Pro 16.2.1 build-18811642

虚拟系统镜像：Ubuntu 16.04.7 desktop amd64

内核版本：4.14.268

### 实验设计

编写字符设备驱动程序，其主要内容包括函数关系绑定、模块的初始化、需要实现的功能、模块的卸载。其中，实现的功能为一个字符缓冲区，可以接收暂存输入的字符串，并支持读取操作。当字符设备加载完毕时，字符缓冲区内字符串为”device started”。

字符设备程序编写较为简单，其中注册模块只需要执行register-chrdev函数获得设备号和判定返回值是否正确；注销模块调用unregister\_chrdev函数将设备号注销即可。读取缓冲区和向缓冲区写入数据需要调用copy\_to\_user和copy\_from\_user函数来实现数据向设备文件的存取。

编写Makefile文件，编译出相关模块，并将其加载到系统中。只需要在对提供的Makefile文件进行修改即可。

编写测试程序对字符设备进行测试。可依次测试各个功能是否正确运行。

## 实验调试

### 实验步骤

1. 编写字符驱动程序。

按照3.3.2中所述，编写出字符设备驱动程序的代码。

1. 编写Makefile文件和进行编译。

实验指导中提供了Makefile文件示例，只需要将其中的文件名改成自己编写的源文件名即可。在存放源文件和Makefile文件的目录下运行make命令即可开始编译，编译获得一.ko文件，即编译出的字符设备模块文件。

1. 装载模块

使用insmod命令装载模块，需在命令前加sudo或直接在su下运行。

1. 分配设备号

使用命令cat /proc/devices查看所有设备，找到自己添加的设备和设备号；如果在源代码中使用printk将设备号输出，也可以使用dmesg命令查看输出的设备号。随后使用mknod命令加入自己的设备即可。

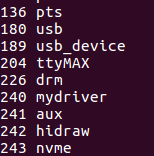


图 3.1 240号为添加的字符设备

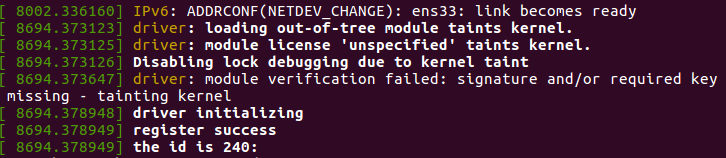


图 3.2 最后三行为使用printk命令输出的信息

1. 编写和运行测试程序

### 实验调试和心得

在编写Makefile文件时，文件的第一行定义了这个模块适用的linux内核版本，实例中采用的是当前使用的内核版本。由于编译模块时需要涉及到内核源码文件，所以需要保证存在该文件夹，否则就会报错。因此可以直接在实验二重启更换内核后，直接在该内核版本下进行模块的编译。

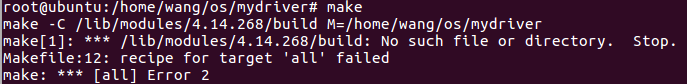


图 3.3 内核源码文件夹不存在时会报错

两次运行测试程序，首次运行时输出了字符缓冲区中的预设内容，第二次运行则输出了第一次输入的内容。可见字符设备和驱动程序运行正确。



图 3.4 字符设备的测试结果

**心得体会：**

本次实验的内容为设备驱动程序的编写和安装，涉及到Makefile文件的编写和使用，是此前从未接触过的内容。但在示例文件和网络资料的的帮助下还是顺利解决了问题。

# 实验四 /proc文件分析

## 实验目的

了解和掌握/proc文件系统的特点和使用方法。

## 实验内容

了解/proc文件的特点和使用方法；

监控系统状态，显示系统中若干部件使用情况；

用图形界面实现系统监控状态。

## 实验设计

### 开发环境

**物理机环境：**

处理器：Intel® Core™ i7-10870H CPU @ 2.20GHz

操作系统：Windows10专业版 21H2 (内部版本19044.1586)

**虚拟机环境：**

虚拟化软件：VMware® Workstation 16 Pro 16.2.1 build-18811642

虚拟系统镜像：Ubuntu 16.04.7 desktop amd64

内核版本：4.14.268

### 实验设计

时间关系，实验四只完成了相对容易的部分。

功能1：直接读取/proc/sys/kernel/hostname中内容即可。

功能2、3：读取/proc/uptime中内容，第一个数据为系统启动时间，第二个数据为系统运行总时间。

功能4：直接读取文件可以/proc/sys/kernel/ostype得到系统类型，读取文件/proc/sys/kernel/osrelease可以得到内核版本号。

功能5：读取文件/proc/cpuinfo，文件列出了所有CPU核心的信息，只需获取其中一个核心的信息即可。其中第5行model name显示了CPU型号和主频。

功能7：要读取所有进程的信息，则需要读取/proc目录下所有名称为数字的文件夹下的stat文件。而要依次读取这样的文件夹，可以利用上学期操作系统实验四的方法，先用opendir函数打开/proc目录，再用readdir函数依次读取/proc目录下的文件夹，若文件夹名为数字，则进入目录，读取其中的stat文件。在stat文件中，第1项为进程pid，第2项为应用程序名，第3项为任务的状态，第4项为ppid，第18项为进程优先级，第23项为进程虚拟地址空间大小。

功能10：此功能不涉及proc文件系统，可调用QT的库函数进行实现。

功能13：可使用system函数打开一个程序。

功能14：可使用system函数执行shutdown命令。

## 实验调试

### 实验步骤

按照4.3.2中所述方法读取信息，利用QT中的label和table等组件实现信息的输出。注意，功能2，3，10需要进行定期刷新。运行并测试功能13、14是否正确工作。

### 实验调试和心得

编译和运行程序，其结果如图所示。经测试，功能13、14均能正确实现。

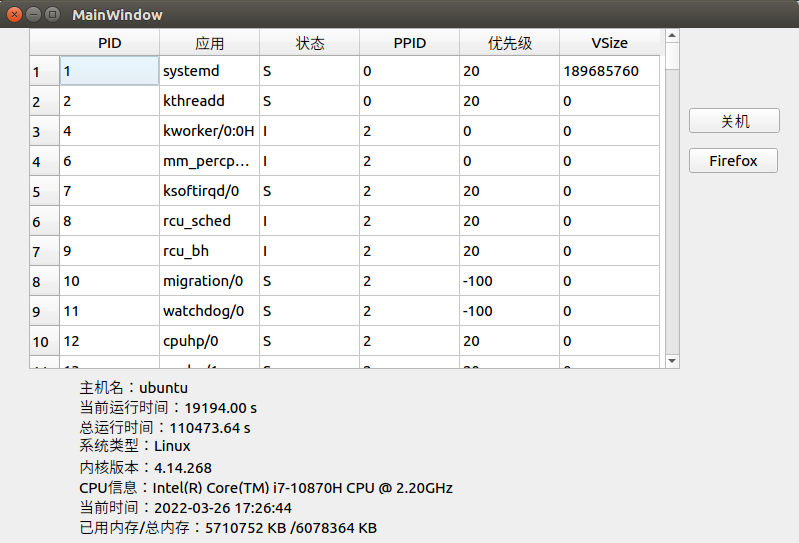


图 4.1 程序的运行效果

心得：本实验的主要难点在同样于图形界面的实现。但是QT提供了图形化的图形界面编译工具和列表、按钮、标签等控件及控制函数，可以方便地实现图形界面和相关的功能。另外，对于系统程序信息的读取，让我复习了上学期所用过的读取文件夹下所有文件的方法，这样做避免了简单按照数字顺序进行搜索的较长时间消耗和超过搜索范围的文件缺失问题。